# 概均質ベクトル空間のゼータ関数の関数等式にあらわれるMATRIXのDETERMINANTについて : REDUCE3.2の利用(概均質ベクトル空間の最近の発展)

AUTHOR(S):

室, 政和

CITATION:

室, 政和. 概均質ベクトル空間のゼータ関数の関数等式にあらわれるMATRIXのDETERMINANTについて : REDUCE3.2の利用(概均質ベクトル空間の最近の発展). 数理解析研究所講究録 1987, 629: 98-116

ISSUE DATE:

1987-06

URL:

http://hdl.handle.net/2433/100007

数理解析研究所講究録
629巻 1987年 98-116

# 概均質ベクトル空間のゼータ関数の関数等式にあらわれる MATRIXのDETERMINANTについて (REDUCE3.2の利用)

高知大学理学部　　室　政和
(Masakazu Muro)

## 0．はじめに

筆者が概均質ベクトル空間上のゼータ関数に関する関数等式などを計算しているときにぶつかる悩みのひとつに、単純ではあるが大量の多項式の演算や行列式を間違えずにこなさなければならない、ということがあります。特に実験的な計算をしているときにはつくづくそう感じます。少し以前まではこのような計算に計算機を利用しようとしても、多項式や有理式を含めた演算をサポートしてくれるような言語は見当たらず、たとえ計算機が利用可能でもそのために自分でプログラムを書いて作らなければならない状態でした。また数式処理というシステムがあるということを知っていてもそれは高価な機械の上で動くもののこと、とうてい我々が手軽に使えるものではない、とあきらめていました。

ところが最近、おもに応用面からの要請でしょうが、パソコンの上でも動く本格的な数式処理システムが市販されはじめました。使いやすいエディタがないなど使い勝手はまだまだですが、とにかく一台備えればけっこうな計算をやってくれます。私達の教室でもＰＣ－９８０１上のＭＳ－ＤＯＳで動く "Reduce on PC" とＰＣ－９８０１に６８０００カードと２メガバイトのメモリを実装して動かす "Stafflisp+Reduce 3.2" を購入することができましたので、さっそく手でやるにはすこし荷が重いと思っておりました計算を実際に試してみました。

今のところ始めたばかりということもあって、これらを使って数学的に意味のあるオリジナルな結果を出すというわけにはいきませんが、それでもおもしろそうな結果を次々と出してきてくれます。特に多項式係数の行列の行列式、さらにその因数分解などをコマンドひとつでやってくれるというのは大変に魅力のあることです。

ここでは、筆者が主に興味を持って研究している、いわゆる「新谷」のゼータ関数の関数等式にあらわれる行列についていろいろと計算を試みた結果を報告したいと思います。実はこのプログラムを書いて計算させたあとで思い出したのですが、[Satake-Faraut] はこの行列が実際に対角化されるということを、ある場合を除いて証明しています。（除かれているのは第１節であげた３種うちの（１．１），１）の場合です。）　それればかりでなく、彼等はどのような行列で変換すれば対角化できるか、

ということまで計算しています。行列はいったん対角化されてしまえば、その行列式を求めることは容易ですし固有値も求まってしまいます。

ですから、ここで計算したことの大部分はすでに証明されていることを計算機で試したという面が強くなってしまったのですが、それでもたとえば[Satake-Faraut]でも除かれている場合などではもう少し工夫をして計算を続ければ対角化などに期待が持てそうな気がしますし、手軽にできるのならこういうことで実験するのも悪くないと思われます。

というわけで、少し変わった話題として、ここにどのような計算をどのようにしてＲＥＤＵＣＥを使ってやったかということをプログラム例をつけて解説したいと思います。計算機にによっておこなう計算自体はめんどうではあっても単純なもので、プログラムには何の工夫もないのですが、こういう機械的な計算になってしまうというところにいくまでには超局所解析の方法を使います。

ＲＥＤＵＣＥを使ってみての感想をいえば、これだけの計算を十分に高速にやってくれるのですから、もっと工夫すればもっと価値のある計算ができそうな気がしますが、今のところそこまではいっていません。いずれこれを使ってもっと確からしい結果が出れば、別の機会に報告したいと思います。

## １．概均質ベクトル空間について

ここでは概均質ベクトル空間についての一般的な説明は避けて、具体的に扱う例をつうじて概均質ベクトル空間を説明しましょう。

まず例として扱う概均質ベクトル空間の定義をできるだけ初等的に与えましょう。Ｖを次のように定義された実数体Ｒ上の有限次元ベクトル空間とします。

（１．１）　１）　$V := Sym_n(R)$
　　＝ｎ×ｎ実対称行列全体のなす空間。

２）　$V := Her_n(C)$
　　＝ｎ×ｎ複素Ｈｅｒｍｉｔｅ行列全体のなす空間。

３）　$V := Her_n(H)$
　　＝ｎ×ｎ四元数Ｈｅｒｍｉｔｅ行列全体のなす空間。

ここで、Ｃは複素数体、Ｈはハミルトンの四元数体をあらわします。これらの空間は実数体Ｒ上のベクトル空間としては各々、

（１．２）　１）　ｎ×（ｎ＋１）／２、

２）　ｎ×ｎ、

3）　$n\times(2n-1)$、

の次元を持つ空間になります。Vの元xは正方行列ですから行列式det（x）を定義することができて、これはV上の多項式になります。もちろん、3）の場合には非可換体上の正方行列ですから、行列式の定義はあらためて与えてやらなければなりませんが、これもふくめて行列式det（x）はV上の実係数多項式になります。いまGL（V）によって、VからVへの線形写像の全体で可逆なもの全体をあらわします。当然この集合は群になっています。特にこの集合のなかで

（1．3）　$G:=\{g\in GL(V);\det(g\cdot x)=\nu(g)\det(x)\}$

で定義される部分集合を考えましょう。ここで$\nu$（g）はG上の実数値関数です。GがGL（V）の部分群になることはすぐにわかりますから$\nu$はGからRへの群の連続な準同型です。そして実際にはGは次のような集合として与えられます。

（1．4）　1）　$G=\{g\in GL_n(R);\ g:\ x\longmapsto gx^tg \text{ for all } x\in V\}$

2）　$G=\{g\in GL_n(C);\ g:\ x\longmapsto gx^tg^c \text{ for all } x\in V\}$

3）　$G=\{g\in GL_n(H);\ g:\ x\longmapsto gx^tg^c \text{ for all } x\in V\}$

ここで、たとえば1）の場合、$GL_n(R)$はn×nの行列式が0でない実数行列をあらわし、その元gの作用は $x\longmapsto gx^tg$ で与えられるものであることをいっています。ここで${}^tg$はgの転置行列をあらわします。同様にして2）および3）の場合も定義されますが、ここでは${}^tg^c$はgの転置行列の各成分の複素共役をとって得られる行列をあらわしています。

（1．1）1），2），3）のいずれの場合もGは連続群になりますのでその単位元を含む連結成分があってそれもまた群になります。それを$G^+$と書きます。すると次の命題が成り立ちます。

命題1．

1）Vは$G^+$の作用で有限個の$G^+$－軌道に分かれる。

2）特に集合$V-\{x\in V;\det(x)=0\}$の各連結成分はひとつの$G^+$－軌道になっている。これらを開軌道と呼ぶ。

3）集合$V-\{x\in V;\det(x)=0\}$は、（n＋1）個の開軌道に分かれる。それらは正および負の固有値の数で特徴づけられる。すなわち、任意の$V-\{x\in V;\det(x)=0\}$の元xは適当な$G^+$の作用によって実対角行列になり、対角成分は正または負の数になる。この正の対角成分の個数と負の対角成分の個数が、それぞれ（i，n－i）となるような元xの集合を$V_i$とする時、これが $V-\{x\in V;$ d

$\det(x)=0\}$ のひとつの連結成分で、$G^+$－軌道となる。

この命題１の２）で言っていることが、Ｖが概均質ベクトル空間であるということの定義です。ここでは（１．１）の１）、２）、３）に扱うものを限っていますので、概均質ベクトル空間という言葉を特に意識する必要はありません。おおざっぱに言えば、ベクトル空間Ｖ上に同次多項式Ｐ（ｘ）があってこれが命題１の２）における$\det(x)$のかわりになって２）における条件を満たすようなＧＬ（Ｖ）の部分群$G^+$があるような場合であれば、これを（正則な）概均質ベクトル空間といっているのです。

このようなとき、集合$V-\{x\in V;P(x)=0\}$の各連結成分は錐(cone)になります。ここでは定義を述べることはできませんが、これらの連結な錐に対してゼータ関数が定義されて、これが概均質ベクトル空間に付随するゼータ関数とよばれるものになります。特にここで例に取り上げた（１．１）の１）、２）、３）の場合は「可換放物型（commutative parabolic type)」として[Muller-Rubenthaler-Schiffmann]によって分類されたもっとも基本的な概均質ベクトル空間になっています。さらに、これらはいわゆる「新谷」のゼータ関数をあたえる概均質ベクトル空間になっています。ここでは、ゼータ関数についての説明は省略して、とくにその関数等式を計算する時にあらわれる群の作用で不変な超関数（これを局所ゼータ関数とよびます。）のフーリェ変換の計算について述べるつもりです。概均質ベクトル空間のゼータ関数については、[Sato-Shintani]、および[Shintani]に詳しく書いてありますのでそれを見てください。

## ２．局所ゼータ関数のフーリェ変換

前節で定義したベクトル空間Ｖとその開軌道の全部の集合$V-\{x\in V;\det(x)=0\}$を考えます。まず、$V-\{x\in V;\det(x)=0\}$は（ｎ＋１）個の連結成分に分かれます。前節の命題で述べたとおり、各々は$G^+$－軌道となります。$V_i(i=0,\dots,n)$を正の固有値がｉ個、負の固有値がｎ－ｉ個であるような元に帰着されるような点の全体としますとこれが$G^+$－開軌道になります。定義より、この開軌道の上では$\det(x)$は０ではないので、この関数の複素巾：

$$(2.1)\quad |P(x)|^s:=|\det(x)|^s$$

は意味を持ちます。そこでまず$V-\{x\in V;\det(x)=0\}$上で

$$(2.2)\quad |P(x)|_i^s:=\begin{cases}|\det(x)|^s & \text{if } x\in V_i\\ 0 & \text{if } x\notin V_i\end{cases}$$

という関数を定義しますと$|P(x)|_i^s$はV－{x∈V；det(x)＝0}上で連続関数になります。さらに{x∈V；det(x)＝0}上では0であるとして$|P(x)|_i^s$を延長しますと、これはまったく一般のs∈Cについては V上の連続関数とはなりませんが、sの実部：Re(s)が十分に大きい時には連続関数になります。したがってf(x)をV上の無限回微分可能な関数で有界な台を持つものとすると、積分：

$$(2.3)\qquad \int |P(x)|_i^s \; f(x)\,dx$$

は、Re(s)が十分おおきなところでは絶対収束してsについて正則な関数になります。そしてs∈C全体へもこれは有理型関数として解析接続できることが証明され、その極の位置はf(x)には依存しません。

そこでsが極の位置ではない時

$$(2.4)\qquad f(x)\longmapsto \int |P(x)|_i^s \; f(x)\,dx$$

という写像を考えますとこれはV上の有界な台を持つ無限回微分可能な関数のなす空間上の線形汎関数になります。このようにして $|P(x)|_i^s$ をそれが必ずしも連続でないところまで（2．6）の形の線形汎関数として定義することができます。これによって $|P(x)|_i^s$という関数をすべての（そこで極ではない）sについて定義することができて、よく知られているように、これがシュワルツ流の超関数の定義の仕方です。ともかくこのようにしてすべての（そこで極ではない）sについて、$|P(x)|_i^s$が超関数として定義されます。

今度は、このようにして定義された超関数のフーリェ変換を考えます。普通の関数f(x)に対しては、

$$(2.5)\qquad f(y)\hat{}:=\int f(x)\exp(-2\pi i\langle x,y\rangle)\,dx$$

とするのがフーリェ変換の定義ですが、これに対して、

$$(2.6)\qquad f(x)\longmapsto\int |P(x)|_i^s \; f(x)\hat{}\,dx$$

という線形汎関数を定義して、これを 超関数$|P(x)|_i^s$のフーリェ変換の定義とします。ここでf(x)がV上の有界な台を持つ無限回微分可能な関数であるからといってそのフーリェ変換f(x)^もそうであるとは限りません。（一般にはそうはなりません。） しかし（2．6）の形の積分はやはり絶対収束しますので、線形汎関数としてはこれはちゃんと定義できます。

このようにしてフーリェ変換を定義して、実際にフーリェ変換を計算することが問題になりますが、$|P(x)|_i^s$は群の作用で不変であると言う性質から、そのフーリェ変換もやはりそうであることが分かります。実はこれが概均質ベクトル空間に付

随するゼータ関数が関数等式を持つ根拠になるのです。超関数として定義された、$|P(x)|_i^s$のフーリェ変換は次の形の公式で与えられることが証明されます。

（２．７）　$|P(x)|_i^{s\,\wedge} = \Sigma_{j=0,\ldots,n} v_{ij}(s)\,|P(x)|_j^{-s-(m/n)}$

ここで$v_{ij}(s)$はｓについての有理型関数、ｍは（１．２）で定義したＶの次元をあらわします。そこで問題は$v_{ij}(s)$を具体的に計算することに帰着されます。この$v_{ij}(s)$を少し書き換えて

（２．８）
$$\begin{aligned} & v_{ij}(s) \\ &= (2\pi)^{-ns-n(v(n-1)+3)/2} \times 2^{-n(n-1)v/4} \\ &\quad \times \Pi_{p=1,\ldots,n}\Gamma(s+1+(v(p-1)/2)) \\ &\quad \times a_{ij}(s) \end{aligned}$$

として$a_{ij}(s)$を定義します。ただし、ここで

（２．９）
$$v := \begin{cases} 1、 & （１．１），１）の場合、 \\ 2、 & （１．１），２）の場合、 \\ 4、 & （１．１），３）の場合、 \end{cases}$$

とします。こうして

（２．１０）　$A(s) := (a_{ij}(s))$

という$a_{ij}(s)$を$(i+1, j+1)$－成分とする$(n+1)\times(n+1)$行列が定義されます。すると各$a_{ij}(s)$は$Q := \exp(\pi i s/2)$と$Q^{-1}$を変数とする多項式になります。そうしてこの$a_{ij}(s)$ならば、機械的に計算することができ、これをREDUCEでプログラムすることもできるようになるのです。

## ３．行列Ａ（ｓ）に関する計算。

フーリェ変換の計算は行列Ａ（ｓ）の計算に帰着されたわけですが、実際にこの計算を実行することはそれほどやさしい問題ではありません。ただここで取り上げた場合にかぎって言えば、その計算のためにさまざまな方法が開発されています。そのうちでももっとも強力であろうとおもわれるのは超局所解析による方法で、これは$|P(x)|_i^s$がみたすホロノミック系とよばれる微分方程式系を利用するもので、ホロノミーダイヤグラムとよばれる図形をを描く必要があるため随分と手間はかかりますが、最終的には機械的な計算に置き換えられますので、すぐれた方法であるといえます。それだけではなく、手間をかけてかいたホロノミーダイヤグラムなどはフー

リェ変換の計算のほかにもさまざまの用途に使えるので、たとえ他の方法で計算されていても、超局所解析の方法できちんと計算しておくことは意味のあることです。ここでは、計算の方法は説明しません。[Muro]を参照してください。

さてこれからＡ（ｓ）の計算に関するREDUCEのプログラム例を３つ紹介します。付録：プログラム例を参照してください。まず最初はＡ（ｓ）の各成分を計算するプログラムです。プログラム例１がそれです。このフログラムをＩＮコマンドでREDUCEに読み込ませてから、COEFFDO(n,v);　と入力します。n,vはそれぞれ数字を入力します。ｖは（２．９）で定義した数字です。そうすると、Ｃ（ｎ，０，ｊ）という配列がＡ（０），．．．，Ａ（ｎ）の線形結合としてあらわされ、その係数が$a_{ij}$（ｓ）をあらわすＱの有理式になります。出力される有理式の変数はＱ，Ｌですがそれぞれ、

（３．１）　　Ｑ：＝ｅｘｐ（πｉｓ／２）、
　　Ｌ：＝ｅｘｐ（πｉ／４）、

をあらわしています。さらにCOEFIL(n);と入力すれば結果がCOEFFというファイルに入ります。そしてSEIRI(n);と入力することによって配列ＣＣ（ｎ，０，ｊ，ｉ）に$a_{ij}$（ｓ）が入ってCOEFF2というファイルに出力されます。

もっともこのようにして計算したところでこれはすでに何人かの人によって計算されているものですから特に目新しいことはありません。ただこのプログラムの目的はもう少し別にあって最後に出てくるＣＣ（ｎ，０，ｊ，ｉ）だけでなく、計算途中の係数もすべて計算することにあるので別に無駄なことをやっている訳ではありません。

次にプログラム例２ですが、これはＡ（ｓ）の行列式を計算しさらにその因数分解をさせるものです。実行の仕方は例によってＩＮコマンドによってプログラムを読み込ませてから、JIKKOU(n,v);と入力します。ｎ，ｖはプログラム例１と同じく数字を入れます。こうすると、GYODET1，およびGYODET2にそれぞれＡ（ｓ）の行列式とそれを因数分解したものが出力されます。こういう計算はREDUCEは得意とみえてｎ＝１０程度までならすぐに結果を出してきます。これから予想される結果は次の通りです。

予想

１）Ａ（ｓ）の行列式は（１．１），１）の場合、

$$\varepsilon(n)\,(Q^4+1)^{a(n)}\,(Q^4-1)^{b(n)}$$

で与えられる。ここで

$$\varepsilon(n) := \begin{cases} 1 & (n \equiv 2 \pmod 4) \\ -1 & (n \equiv 2 \pmod 4) \end{cases}$$

$$a(n) := [n/2] \times ([n/2]+1)$$

$$b(n) := [(n+1)/2]^2$$

2）A（s）の行列式は（1．1），2），3）の場合、

$$((Q^2+1)(Q+1)(Q-1))^{n(n+1)/2}$$

で与えられる。

このうち2）の結果はすでに[Satake-Faraut]によって証明されているので予想ではありません。1）はだれかがすでに証明をもっていて発表していないという可能性はありますが筆者自身はまだ証明できていません。

最後にプログラム例3。これはプログラム例2とほとんど変わりがありません。ただA（s）の行列式を計算するかわりにA（s）－z$I_n$の行列式を計算してそれを因数分解して出力させるものです。首尾よく因数分解すればA（s）の固有値が求まるのでこれでA（s）を対角化することができることになります。ところがここにいたってどうもうまく計算が進まなくなりました。REDUCEは虚数単位Iが入っていたり、多変数だったりすると、どうも因数分解を完全にやってはくれないようです。ここではREDUCEで出した結果にさらに手を加えたものを以下に書きます。

計算結果

（1．1），1）の場合A（s）－z$I_n$の行列式は次のように因数分解される。

1） n＝1 のとき、

$$(Qz+i(1-Q^2))(Qz+i(1+Q^2))/Q^2$$

2） n＝2 のとき、

$$(Q^2z-i(1-Q^4))(Q^2z-i(1+Q^4))$$
$$\times(Q^2z+(1+Q^4))/Q^6$$

3） n＝3 のとき、

$$(Q^6z^2-Q^{12}+Q^8+Q^4-1)^2/Q^{12}$$

4） n＝4 のとき、

$$(Q^4z-i(1+Q^4)^2)(Q^4z-i(1+Q^4)(1-Q^4))$$
$$\times(Q^4z+(1+Q^4)^2)(Q^4z+(1-Q^4)^2)$$
$$\times(Q^4z+(1+Q^4)(1-Q^4))/Q^{20}$$

5） n＝5 のとき、

$$(Q^5z-i(Q^4+1)(Q^4-1)(Q^2+1))$$
$$\times(Q^5z+i(Q^4+1)(Q^4-1)(Q^2+1))^2$$
$$\times(Q^5z-i(Q^4+1)(Q^4-1)(Q^2-1))^2$$
$$\times(Q^5z+i(Q^4+1)(Q^4-1)(Q^2-1))/Q^{30}$$

筆者の持っているREDUCEではｎ＝７までは計算してくれるのですが、複雑なので以下省略します。ｎ＝１５までぐらい計算できれば予測がつくと思うのですが〝これだけの結果からはなんともいえません。また（１．１）の２）、３）の場合はすでに[Satake-Faraut]が対角化していますのでここでは結果を書きません。

## ４．終わりに

ここでプログラム例をあげてまで説明してきましたが、ようやく個人でも使えるようになった数式処理システムをちょっと試してみた、という以上の内容はありません。今のところ予測のようなものをしたり、検算をしたりといったことには使えそうな気がしますが、それでもそれで証明にすぐ結びつくものではありません。その上メモリの制約やプログラミングの環境など現在のシステムはまだまだ問題も多くあります。そして、ハードウェアやソフトウェアの技術的な問題はいずれ解決されるとしても、これをうまく使いこなす技術とか、数学の問題をいかに計算機のうえにのせるかといった工夫のようなものは、ひとりひとりが試行錯誤しながらみつけていくしかないようで、それは結局機械が解決してくれる問題ではなさそうです。

とはいえパソコンの上でこれだけ高速の計算をやってくれるとは思っていませんでしたという感想もしるしておかなければ、作成者（システムの作成者およびパソコンのうえにインプリメントしてくれた人）に失礼と言うものでしょう。筆者の場合、ほんの実験的な計算しかしておりませんし、プログラムも正直に手順を書いただけの何の工夫もないものですが、本来、因数分解などは出てきた結果を見ながら因数を予測してすすめていったほうが良いでしょうし、そうしてこそ数式処理が真の数学のアシスタントシステムになれるだろうと思います。

最後に参考までに筆者の使ったシステムを書いておきます。まず、

１）ＲＥＤＵＣＥｏｎＰＣ　（ＢＵＧ）

ですが、これはＮＥＣのＰＣ－９８０１に６１２キロバイトのメモリを実装して使うものです。これはＯＳがＭＳ－ＤＯＳですのでプログラムを書く時、ＭＳ－ＤＯＳのエディタがつかえます。ＲＥＤＵＣＥは計算の結果をすべて覚えていますので、それを御破算にして新しい計算をやるときにはシステムを抜け出て、あらためてＲＥＤＵＣＥを読み込むといったことをやる必要があります。特にプログラムをテストランさせるときなど何度もＲＥＤＵＣＥとエディタの間を往復しなければなりません。そこでＲＥＤＵＣＥとエディタをハードディスクに置いておき、これらを迅速に往復できるようにして使いました。さらにメモリが足りないときには、

２）Ｓｔａｆｆｌｉｓｐ＋ＲＥＤＵＣＥ３．２　（ＢＵＧ）

に移ります。筆者の場合、ＰＣ－９８０１に６８０００のボードと２メガバイトのメモリ（カノープス電子）を装着し、INTERDISK2（メガソフト）でＭＳ－ＤＯＳ上のプログラムをＣＰ／Ｍに移し、２）のシステムをたちあげて走らせるという使い方をしました。特に因数分解をするときなど、１）のシステムではメモリの関係上まず無理なので２）のシステムを使わざるをえませんが、２）のシステムにはスクリーンエディタがなくて非常に使いづらいので上のようなやりかたをしたわけです。

## 参考文献

[Muller-Rubenthaler-Schiffmann] Muller,I.,Rubenthaler,H. and Schiffmann,G., Structure des espaces prehomogenes associes a certaines algebres de Lie graduees, Math.Ann., 274, (1986) ,95-123.

[Muro] Muro,M., Microlocal analysis and calculations on some relatively invariant hyperfunctions related to zeta functions associated with the vector spaces of quadratic forms , Publ. RIMS, Kyoto Univ. 22 (1986), 395-463.

[Satake-Faraut] The functional equation of zeta distributions associated with formally real Jordan algebras, Tohoku Math. J. , 36 (1984), 469-482.

[Sato-Shintani] Sato,M and Shintani,T., On zeta functions associated with prehomogeneous vector spaces, Ann. of Math., 100 ,(1974), 131-170.

[Shintani] Shintani,T, On zeta functions associated with the vector space of quadratic forms, J. Fac. Sci., Univ. of Tokyo, 22,(1975),25-65.

付録：プログラム例

## プログラム例 1

```
% COEFFDO関数とCOEFIL関数及びKAKUNIN関数の定義。さらに加えてここでは出力をINで
%読み込めるようにする。
%
%COEFFDO関数の実行ファイル:SETTEI関数を実行した後COEFF1関数に移りひとつだけ値
% 返して終る
%
PROCEDURE COEFFDO(N,V);
BEGIN
N1:=N;
V1:=V;
SETTEI(N1,V1);
COEFF1();
RETURN C(N1,0,0)
END;
%
%SETTEI関数の定義: 定数や配列を設定する
%
PROCEDURE SETTEI(N,V);
BEGIN
CLEAR C,D;
ARRAY C(N,N,N);% 主係数の配列
ARRAY D(N,N,N);% 補助係数の配列
% Lを虚数単位の平方根とする
LET L**4=-1;
%変換係数Fの設定
CLEAR F11,F12,F21,F22;
OPERATOR F11,F12,F21,F22;
FOR ALL J,K LET
F11(J,K)=(L**(-2*K*V-2))*Q**(-1),
F12(J,K)=(L**(2*K*V+2))*Q,
F21(J,K)=(L**(2*(J-K)*V+2))*Q,
F22(J,K)=(L**(-2*(J-K)*V-2))*Q**(-1);
% 最初の係数にA(I)を代入する
CLEAR A;
OPERATOR A;
P:=0;
REPEAT <<C(0,P,0):=A(P);P:=P+1>> UNTIL P=N+1
END;
%
%COEFF1関数の定義: I=0からN1までCOEFF2を繰り返す
%
PROCEDURE COEFF1;
BEGIN
FOR I:=0:N1-1 DO COEFF2(I)
END;
%
%COEFF2関数の定義: IをFIXしておいてCOEFF3をJについて繰り返す
%
PROCEDURE COEFF2(I);
BEGIN
I1:=I;
```

```
FOR J:=1:N1-I1 DO COEFF3(I1,J)
END;
%
%COEFF3関数の定義: I,JをFIXしておいてKについて代入を繰り返す
%
PROCEDURE COEFF3(I,J);
BEGIN
I2:=I;
J2:=J;
FOR K:=0:I2 DO <<C(I2+1,J2-1,K+1):=F11(I2,K)*C(I2,J2,K)+F12(I2,K)*C(I2,J2-1,K);
D(I2+1,J2-1,K):=F21(I2,K)*C(I2,J2,K)+F22(I2,K)*C(I2,J2-1,K)>>;
C(I2+1,J2-1,0):=F21(I2,0)*C(I2,J2,0)+F22(I2,0)*C(I2,J2-1,0)
END;
%
%以上でCOEFFDO関数の定義は終り
%
%
%COEFIL(N)関数の定義: C(*,*,*)をすべてCOEFFファイルに出力する
%特にCOEFIL1関数を実行してCOEFFファイルに出す
%
PROCEDURE COEFIL(N);
BEGIN
N1:=N;
OFF NAT;
OUT COEFF;
FOR I:=0:N1 DO COEFIL1(N1,I);
WRITE "%END OF THE COEFFICIENTS FOR N=",N1;
WRITE ";END";
SHUT COEFF;
ON NAT
END;
%
%COEFIL1関数の定義:COEFIL2関数の実行
%
PROCEDURE COEFIL1(N,I);
BEGIN
N2:=N;
I2:=I;
FOR J:=0:N2-I2 DO COEFIL2(J)
END;
%
%COEFIL2関数の定義: 結果をファイルに書く
%
PROCEDURE COEFIL2(J);
BEGIN
J1:=J;
FOR K:=0:I2 DO WRITE "C(",I2,",",J1,",",K,"):=",C(I2,J1,K)
END;
%
%以上でCOEFIL関数の定義は終り
%
%KAKUNIN関数の定義：CもDも同じ値になっていることを確かめる
%
PROCEDURE KAKUNIN(N);
BEGIN
N1:=N;
```

```
IF N1<=1 THEN WRITE "NOT NECESSARY" ELSE
FOR I=2:N1 DO KAKUNIN1(N1,I)
END;
PROCEDURE KAKUNIN1(N,I);
BEGIN
N2:=N;
I2:=I;
FOR J:=0:N2-I2 DO KAKUNIN2(J)
END;
PROCEDURE KAKUNIN2(J);
BEGIN
J1:=J;
FOR K:=0:I2-1 DO IF C(I2,J1,K)-D(I2,J1,K)=0 THEN
WRITE "C",N1,"(",I2,",",J1,",",K,"):=OK!!"
ELSE WRITE "C",N1,"(",I2,",",J1,",",K,"):=ERROR!! SONO SA:=",C(I2,J1,K)-D(I2,J1,K)
END;
%COEFFファイルに入っているデータを読みだして整理する関数。SEIRI関数と呼びCOEFF
%のデータを並びかえCOEFFN2 ファイルに出力する。
PROCEDURE SEIRI(N);
BEGIN
CLEAR C,CC,D;
N1:=N;
ARRAY C(N1,N1,N1);
ARRAY CC(N1,N1,N1,N1);
IN "COEFF";
FOR S:=0:N1 DO <<A(S):=X**S>>; %A(S)をX**Sに置き換えて多項式みたいにする。
FOR I:=0:N1 DO  SEIRI1(I); %SEIRIの手続きへいけ!!
CLEAR C;
OFF NAT;
OUT COEFFN2;
FOR I:=0:N1 DO  KAKIKOMI1(I); %KAKIKOMIの手続きへ行け!!
WRITE "%END OF THE COEFFICIENTS FORN=",N1;
WRITE ";END";
SHUT COEFFN2;
ON NAT;
END;
%SEIRIの手続き。
PROCEDURE SEIRI1(I);
BEGIN
I1:=I;
FOR J:=0:N1-I1 DO  SEIRI2(I1,J)
END;
PROCEDURE SEIRI2(I,J);
BEGIN
I1:=I;
J1:=J;
FOR R:=0:I1 DO  SEIRI3(I1,J1,R)
END;
PROCEDURE SEIRI3(I,J,R);
BEGIN
I1:=I;
J1:=J;
R1:=R;
BUNBO:=DEN(C(I1,J1,R1));
N2:=DEG(C(I1,J1,R1)*BUNBO,X);
ARRAY W(N2);
```

```
COEFF(C(I1,J1,R1)*BUNBO,X,W);
FOR S:=0:N2 DO <<CC(I1,J1,R1,S):=W(S)/BUNBO>>;
CLEAR W,BUNBO
END;
%KAKIKOMIの手続き。
PROCEDURE KAKIKOMI1(I);
BEGIN
I1:=I;
FOR J:=0:N1-I1 DO  KAKIKOMI2(I1,J)
END;
PROCEDURE KAKIKOMI2(I,J);
BEGIN
I1:=I;
J1:=J;
FOR R:=0:I1 DO  KAKIKOMI3(I1,J1,R)
END;
PROCEDURE KAKIKOMI3(I,J,R);
BEGIN
I1:=I;
J1:=J;
R1:=R;
FOR S:=0:N1 DO WRITE "CC(",I1,",",J1,",",R1,",",S,"):=",CC(I1,J1,R1,S)
END;
END;
```

（プログラム例１、終わり）

## プログラム例２

```
%
%ここではMATRIXのDETERMINAT(GYODET1)を計算して因数分解(GYODET2)する。
%
PROCEDURE JIDOU(U);
BEGIN
U1:=U;
FOR I:=1:8 DO JIDOU2(I)
END;
PROCEDURE JIDOU2(V);
BEGIN
FOR J:=1:U1 DO JIKKOU(J,V)
END;
%
%実行関数: JIKKOUの定義。
%
PROCEDURE JIKKOU(N,V);
BEGIN
N1:=N;
V1:=V;
DO COEFFDO(N1,V1);
DO SEIRI(N1);
DO GYO(N1)
END;
%
%COEFFDO関数の定義:SETTEI関数を実行した後COEFF1関数に移る。
%
PROCEDURE COEFFDO(N,V);
```

```
BEGIN
CLEAR X;
N1:=N;
V1:=V;
SETTE1(N1,V1);
COEFF1();
END;
%
%SETTE1関数の定義: 定数や配列を設定する
%
PROCEDURE SETTE1(N,V);
BEGIN
CLEAR C;
ARRAY C(N,N,N);% 主係数の配列
% Lを虚数単位の平方根とする
LET L**4=-1;
%変換係数Fの設定
CLEAR F11,F12,F21,F22;
OPERATOR F11,F12,F21,F22;
FOR ALL J,K LET
F11(J,K)=(L**(-2*K*V-2))*Q**(-1),
F12(J,K)=(L**(2*K*V+2))*Q,
F21(J,K)=(L**(2*(J-K)*V+2))*Q,
F22(J,K)=(L**(-2*(J-K)*V-2))*Q**(-1);
% 最初の係数にX**Iを代入する
P:=0;
REPEAT <<C(0,P,0):=X**P;P:=P+1>> UNTIL P=N+1
END;
%
%COEFF1関数の定義: I=0からN1までCOEFF2を繰り返す
%
PROCEDURE COEFF1;
BEGIN
FOR I:=0:N1-1 DO COEFF2(I)
END;
%
%COEFF2関数の定義: IをFIXしておいてCOEFF3をJについて繰り返す
%
PROCEDURE COEFF2(I);
BEGIN
I1:=I;
FOR J:=1:N1-I1 DO COEFF3(I1,J)
END;
%
%COEFF3関数の定義: I,JをFIXしておいてKについて代入を繰り返す
%
PROCEDURE COEFF3(I,J);
BEGIN
I2:=I;
J2:=J;
FOR K:=0:I2 DO <<C(I2+1,J2-1,K+1):=F11(I2,K)*C(I2,J2,K)+F12(I2,K)*C(I2,J2-1,K);
C(I2+1,J2-1,0):=F21(I2,0)*C(I2,J2,0)+F22(I2,0)*C(I2,J2-1,0)>>
END;
%
%以上でCOEFFDO関数の定義は終り
%
```

```
%SEIRI関数の定義: Cのデータを並びかえてCCに代入。
%
PROCEDURE SEIRI(N);
BEGIN
CLEAR CC ;
N1:=N;
ARRAY CC(N1,N1);
SEIRI2() ;
CLEAR C
END;
%
%SEIRIの手続き。
%
PROCEDURE SEIRI2;
BEGIN
FOR R:=0:N1 DO  SEIRI3(R)
END;
PROCEDURE SEIRI3(R);
BEGIN
CLEAR BUNBO,W;
R1:=R;
BUNBO:=DEN(C(N1,0,R1));
N2:=DEG(C(N1,0,R1)*BUNBO,X);
ARRAY W(N2);
COEFF(C(N1,0,R1)*BUNBO,X,W);
FOR S:=0:N2 DO <<CC(R1,S):=W(S)/BUNBO>>;
CLEAR W,BUNBO
END;
%
%GYO関数の定義:DETERMINANTの計算。
%
PROCEDURE GYO(N);
BEGIN
CLEAR Y, GYODET, GYOATO;
N1:=N;
MATRIX Y(N1+1,N1+1);
FOR I:=0:N1 DO << DAINYU(I)>> ;
CLEAR CC;
GYODET:=DET(Y);
WRITE "GYODET1(",n1,",",V1,")=",GYODET;
ON FACTOR;
WRITE "GYODET2(",n1,",",V1,")=",GYODET;
OFF FACTOR;
WRITE "NEXT!!"
END;
PROCEDURE DAINYU(I);
BEGIN
I1:=I;
FOR J:=0:N1 DO <<Y(I1+1,J+1):=CC(I1,J)>>
END;
END;
```

（プログラム例２、終わり）

# プログラム例３

```
%
%DETERMINANTの対角化を求めるためその特性方程式を計算する。その２。
%
%自動実行関数:JIDOU(U)の定義。
%
PROCEDURE JIDOU(U);
BEGIN
U1:=U;
FOR I:=1:8 DO JIDOU2(I)
END;
PROCEDURE JIDOU2(V);
BEGIN
FOR J:=1:U1 DO JIKKOU(J,V)
END;
%
%実行関数: JIKKOUの定義。
%
PROCEDURE JIKKOU(N,V);
BEGIN
N1:=N;
V1:=V;
DO COEFFDO(N1,V1);
DO SEIRI(N1);
DO GYO(N1)
END;
%
%COEFFDO関数の定義:SETTEI関数を実行した後COEFF1関数に移る。
%
PROCEDURE COEFFDO(N,V);
BEGIN
N1:=N;
V1:=V;
SETTEI(N1,V1);
DO COEFF1()
END;
%
%SETTEI関数の定義: 定数や配列を設定する
%
PROCEDURE SETTEI(N,V);
BEGIN
N1:=N;
CLEAR C;
ARRAY C(N1,N1,N1);% 主係数の配列
% Lを虚数単位の平方根とする
LET L**2=1;
%変換係数Fの設定
CLEAR F11,F12,F21,F22;
OPERATOR F11,F12,F21,F22;
FOR ALL J,K LET
F11(J,K)=(L**(-2*K*V-2))*Q**(-1),
F12(J,K)=(L**(2*K*V+2))*Q,
F21(J,K)=(L**(2*(J-K)*V+2))*Q,
F22(J,K)=(L**(-2*(J-K)*V-2))*Q**(-1);
```

```
% 最初の係数にX**Iを代入する
C(0,0,0):=1;
P:=1;
REPEAT <<C(0,P,0):=X**P;P:=P+1>> UNTIL P=N1+1
END;
%
%COEFF1関数の定義: I=0からN1までCOEFF2を繰り返す
%
PROCEDURE COEFF1;
BEGIN
FOR I:=0:N1-1 DO COEFF2(I)
END;
%
%COEFF2関数の定義: IをFIXしておいてCOEFF3をJについて繰り返す
%
PROCEDURE COEFF2(I);
BEGIN
I1:=I;
FOR J:=1:N1-I1 DO COEFF3(I1,J)
END;
%
%COEFF3関数の定義: I,JをFIXしておいてKについて代入を繰り返す
%
PROCEDURE COEFF3(I,J);
BEGIN
I2:=I;
J2:=J;
FOR K:=0:I2 DO
<<C(I2+1,J2-1,K+1):=F11(I2,K)*C(I2,J2,K)+F12(I2,K)*C(I2,J2-1,K)>>;
C(I2+1,J2-1,0):=F21(I2,0)*C(I2,J2,0)+F22(I2,0)*C(I2,J2-1,0)
END;
%
%以上でCOEFFDO関数の定義は終り
%
%SEIRI関数の定義: Cのデータを並びかえてCCに代入。
%
PROCEDURE SEIRI(N);
BEGIN
CLEAR CC ;
N1:=N;
ARRAY CC(N1,N1);
SEIRI2() ;
CLEAR C
END;
%
%SEIRIの手続き。
%
PROCEDURE SEIRI2;
BEGIN
FOR R:=0:N1 DO  SEIRI3(R)
END;
PROCEDURE SEIRI3(R);
BEGIN
CLEAR BUNBO,W;
R1:=R;
BUNBO:=DEN(C(N1,0,R1));
```

```
N2:=DEG(C(N1,0,R1)*BUNBO,X);
ARRAY W(N2);
COEFF(C(N1,0,R1)*BUNBO,X,W);
FOR S:=0:N2 DO <<CC(R1,S):=W(S)/BUNBO>>;
CLEAR W,BUNBO;
END;
%
%GYO関数の定義:DETERMINANTの計算。
%
PROCEDURE GYO(N);
BEGIN
CLEAR Y, GYODET, GYOATO;
N1:=N;
MATRIX Y(N1+1,N1+1);
FOR I:=0:N1 DO << DAINYU(I)>> ;
FOR I:=1:N1+1 DO <<Y(I,I):=Y(I,I)-Z>>;
CLEAR CC;
GYODET:=DET(Y);
ON FACTOR;%ここから因数分解する。
WRITE "GYODET(",N1,",",V1,")=",GYODET;
OFF FACTOR;
WRITE "NEXT!!"
END;
PROCEDURE DAINYU(I);
BEGIN
I1:=I;
FOR J:=0:N1 DO <<Y(I1+1,J+1):=CC(I1,J)>>
END;
END;
```

（プログラム例３、終わり）